Animated Edge Textures in Node-Link Diagrams: a Design Space and Initial Evaluation

Hugo Romat, Caroline Appert, Benjamin Bach, Nathalie Henry-Riche, Emmanuel Pietriga

Animated Edge Textures in Node-Link Diagrams: a Design Space and Initial Evaluation

Hugo Romat[2,1] **Caroline Appert**[1] **Benjamin Bach**[3] **Nathalie Henry-Riche**[4] **Emmanuel Pietriga**[1]
[1]Univ. Paris-Sud, CNRS, INRIA, Université Paris-Saclay, France
[2]TecKnowMetrix Voiron, France
[3]University of Edinburgh Edinburgh, UK
[4]Microsoft Research Seattle, USA

ABSTRACT
Network edge data attributes are usually encoded using color, opacity, stroke thickness and stroke pattern, or some combination thereof. In addition to these static variables, it is also possible to animate dynamic particles flowing along the edges. This opens a larger design space of animated edge textures, featuring additional visual encodings that have potential not only in terms of visual mapping capacity but also playfulness and aesthetics. Such animated edge textures have been used in several commercial and design-oriented visualizations, but to our knowledge almost always in a relatively *ad hoc* manner. We introduce a design space and Web-based framework for generating animated edge textures, and report on an initial evaluation of particle properties – particle speed, pattern and frequency – in terms of visual perception.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): Graphical user interfaces (GUI)

Author Keywords
Node-link diagrams; Animation; Particles; Networks; Trees.

INTRODUCTION
The amount of literature on graph visualization is considerable [6, 26, 59]. Beyond research on layout algorithms for node-link representations, the community has explored several aspects pertaining to the visual encoding of data attributes on nodes [63] and links [34, 31]. Focusing on links only, data attributes are usually encoded using the following visual variables: color, opacity, stroke thickness and stroke pattern. In the case of dynamic visualizations, this relatively limited set of variables can be augmented using animations [12].

Animating directed edges in node-link diagrams is typically achieved by having sequences of small glyphs, which we will call *particles*, dynamically move along the links [2]. The visual effect is that of an *animated texture* applied to links. In some cases, there is a one-to-one correspondence between particles and entities actually traversing the network, as for instance in the case of traffic visualizations where each particle represents one vehicle [11, 50]. In other cases, there is no such correspondence: the particles are elements of a cyclic animated texture, whose visual properties (pattern, speed, frequency – see Figure 1) encode more abstract data attributes. Examples of the latter include visualizations of telecommunication networks (attributes such as, *e.g.*, failure rate, lag, bandwidth) or import/export of goods between countries [13].

While several network visualizations feature animated edge textures on directed links (see, *e.g.*, [10, 13, 14, 32, 36, 46]), they do so in a relatively *ad hoc* manner, often lacking a design rationale. Yet, the amount of work leveraging animated edge textures, beyond encoding edge data attributes, hints at their potential. They can help visually relate groups of consecutive edges [8] or guide visual search by highlighting specific edges [60]. They can also serve as a means to illustrate dynamic propagation [1], or contagion [58], through networks. These diverse yet sporadic uses of animated edge textures for encoding information in graphs and networks call for a more systematic exploration of their design space.

This paper seeks to provide a deeper understanding of the perceptual quality of animated edge textures, as well as to explore more systematically their design space. After providing a motivation for our work, we introduce a model and associated Web-based framework for generating animated edge textures, and illustrate its capabilities using different examples of visual mappings. We then report on an initial evaluation of particle properties in terms of visual perception. The results suggest that the three motion variables defining animated edge textures - speed, frequency and pattern (Figure 1) can be used

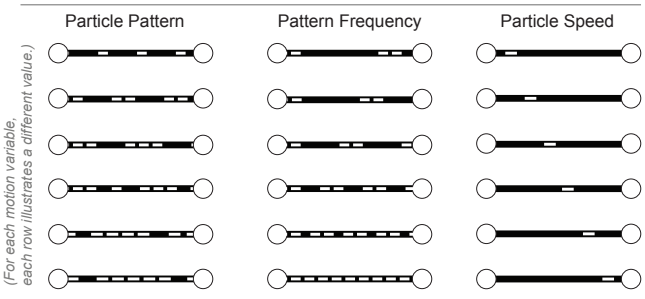

Figure 1. The three variables defining the dynamic behavior of particles in animated edge textures: the *particle pattern*, or rhythm, is the sequence of particles that gets repeated cyclically; the *pattern frequency* corresponds to the firing frequency of the particle sequence; the *particle speed* corresponds to the velocity of particles on screen.

Hugo Romat, Caroline Appert, Benjamin Bach, Nathalie Henry-Riche & Emmanuel Pietriga. Animated Edge Textures in Node-Link Diagrams: a Design Space and Initial Evaluation. In *CHI '18: Proceedings of the 36th Annual ACM Conference on Human Factors in Computing Systems*, 13 pages, ACM, April 2018.

https://doi.org/10.1145/3173574.3173761

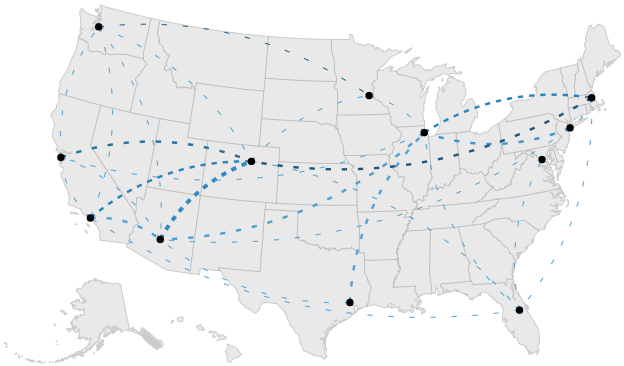

Figure 2. Mapping three data attributes on three static visual variables.

individually for encoding data attributes. We conclude with a discussion of the limitations of this first study, and an outline of the next steps to pursue this line of research.

BACKGROUND AND MOTIVATION

The number of visual variables available for edge attribute encoding in node-link diagrams is limited, and while there are opportunities for combining those variables, they are not completely orthogonal. For instance, color and opacity will, to some extent, mutually interfere. Stroke width, color and opacity will also all influence the saliency of an edge. The practical range of values for these variables is also fairly limited. For instance, wide strokes are likely to cause visual occlusion of elements in the background, or of other edges in dense graphs. Beyond the question of efficiency of the encoding in terms of visual perception, issues related to the aesthetics of the visualization can quickly arise, that go beyond pure considerations of graph layout [62].

Taking the example of a statistical dataset about domestic flights in the USA,[1] the number of featured edge attributes far exceeds the number of visual variables. As illustrated in Figure 2, the number of airlines operating on a route could be encoded using the stroke width; the daily number of flights could be encoded using a stroke pattern made of dashes whose spacing is a function of that value; and the daily number of passengers on a given route could be encoded using color brilliance. But encoding more attributes would be challenging.

By gaining a better understanding of what other visual variables can be used to encode data in node-link diagrams, our goal is *not* to find *better* visual variables, but rather to identify *alternative* encodings, widening the design space of visual mappings in node-link diagrams, in the same spirit as Henry Riche *et al.* [31], who explored the design space of link curvature to represent attributes such as edge directionality. In this work we focus on motion, which has been identified by Bartram, Ware and colleagues as *"hold[-ing] promise [...] as a perceptually efficient display dimension"* [4].

Coming back to the above example about air traffic statistics, the introduction of motion as a means to encode data widens the space of possibilities in two ways. It can either be seen as a way to encode additional data attributes: for instance, the average speed on a route can be encoded using particle speed. Or it can be seen as a way to encode some data attributes using a different visual variable deemed better suited: for instance, mapping the daily number of passengers to particle speed can give a sense of route "throughput" that color brilliance might not convey as effectively. Motion-based encodings can also provide alternatives when visual variables such as color are not available, for example when the underlying layers in the visualization already make use of color to encode other data, or on electronic ink (monochrome) displays.

Motion has potential as a means to encode edge data attributes in graphs, and more specifically in *directed* graphs, as motion along an edge explicitly suggests a specific orientation of that edge. But possible usages of motion in network visualization need to be better understood. Our goal is to define a design space of motion-based data encoding in node-link diagrams, and to enable the in-depth empirical assessment of their characteristics in terms of perception. To this end, we introduce a framework called Animated Edge Textures. Before introducing this framework, we discuss findings from the literature that are relevant to our research.

Motion in Human-Computer Interaction

While animations, including motion, should be used with caution [57], there are situations where they have proven useful. Studies about the potential of motion in HCI date back to the early 1990s with seminal work such as Ware *et al.*'s investigation of motion as a means to attract user attention [61]. Motion is also useful in scientific visualization [35], as the processes visualized often involve the dimension of time [22]. Very early, Bartram started investigating motion as an *"abstractly codable dimension in its own right"* [3], a direction explored in other projects [4, 5, 39, 40], and in which we situate our own work.

In information visualization, animations might not be effective as an exploration aid, but can be powerful tools for presentation [22]. Motion can be useful for filtering and brushing [4, 19, 60]. It can help emphasize spatial relationships, explain functionality, illustrate causality [45, 5]. It is also used extensively in flow visualizations [28, 35, 44] and representations of other scientific data such as cosmological particles [27].

Motion has been observed to evoke affect [21] and has an impact on the perceived aesthetic qualities of a visualization. This likely plays a role in the use of particle-based animations in some design-oriented visualizations [13, 14, 32, 46, 49].

Motion Perception

The body of literature on motion perception in experimental psychology is very large [52]. While it is not our intent to report them exhaustively in this section, we highlight key findings on motion perception that makes this encoding compelling to explore for information visualization researchers. We also motivate the need for perception studies closer to visualization applications.

Early research demonstrated the ability of simple motions to communicate complex or nuanced behaviors [30]. A crucial insight for information visualization is that motion belongs

[1] Obtained from https://www.transtats.bts.gov.

to the limited set of visual encodings that is perceived preattentively [56], *i.e.*, detected before focused attention. Motion can make moving shapes naturally "pop out" in visualizations. Orban *et al.* [48] observed that different motion velocities are easier to discriminate in the central visual field, the just-noticeable difference in velocity increasing as eccentricity increases, particularly so for low reference velocities.

The perception of motion in conjunction with other encodings such as color has been studied in prior work showing, for example, that a conjunctive search for motion and color is serial rather than parallel [47], whereas a conjunctive search for motion and form is parallel [43]. Questions related to conjunction in guided search have been further studied by Driver and colleagues. They observed that it can be parallel under some circumstances, but that out-of-phase motion across elements had a significant negative impact on search performance [17]. They also observed that the search for a target with a different orientation was faster among moving distractors than stationary ones, provided the target's orientation is not too close to that of non targets [16].

Beyond the Gestalt law of common fate, which states that elements tend to be perceived as a group if they move together, another interesting observation is that moving objects can capture attention [23]. Scimeca and Franconeri [51] highlight the main findings related to tracking multiple objects (or groups of objects) in motion. Multiple studies have investigated our ability to detect changes in direction [53], speed [42, 37] or both [25, 33]. The role that the color and luminance of elements play in the perception of motion direction and velocity has also been studied extensively, as summarized by Weiskopf [64].

The above psychophysiological-level findings can guide and inform the research and design of novel visualization techniques, but they cannot, alone, answer higher-level questions related to the perceptual efficiency of motion-based visual encodings in node-link diagrams. Indeed, the use of motion on graph edges is quite specific. The considered elements (particles) only move along predefined visual paths. Particles that belong to the same edge get perceived as a group not only because they have a common fate but also because they visually materialize a first-class entity: edges. The nature of the movement and the number of different directions, considering that edges have different orientations and that they can be curved, is very particular compared to the conditions evaluated in the above-cited studies. Empirical studies that focus on these conditions are necessary to get a clear understanding, and to quantify the limits, of motion-based data encodings in tree and network visualization.

ANIMATED EDGE TEXTURES

Our model of animated edge textures consists of three main motion-related variables describing the behavior of particles flowing along links, as illustrated in Figure 1: the *pattern* of particles, which can be seen as the rhythm at which they get fired; the *frequency* of this pattern, and finally, the *speed* of particles along the link. Each variable can be mapped to different data attributes. They can also be combined with other *visual variables* that define the appearance of both the particles (*e.g.*, their color, their shape) and of the link itself to encode additional edge data. In this section, we describe the design space of animated edge textures, and introduce an API for mapping data attributes to visual variables in this design space, along with a prototype implementation in WebGL.

Model

The simplest way to animate particles along an edge would be to define a repeating stroke pattern, as one can do in vector graphics editors such as Adobe Illustrator, and have this repetition of the pattern animate by gradually increasing its start offset. However, this approach is limited in terms of expressive power. It also fails to expose all motion variables for direct mapping with data attributes. We seek to strike a balance between simplicity of the model, expressive power, and ease of mapping between data attributes and visual variables.

Our model is based on the following core concepts:

- The *link* itself, which encodes the path geometry of the corresponding edge (for a given layout). The link can be seen as a tunnel through which the particles flow. The visual appearance of that tunnel is defined by the static visual variables usually associated with links in a node-link diagram: color, opacity, stroke width.[2]
- The *particle emitter* fires particles from the source node of an edge according to the particle pattern and pattern frequency. A particle emitter defines the properties of particles it fires: their shape, color, size, and speed.
- The *particles* themselves, which are the individual glyphs that flow along the link.

Two additional concepts are introduced to enable more elaborate motion-based encodings:

- *Gates*, placed at arbitrary positions along an edge, can change one or more variables of the particles flowing along that edge, including their color, opacity, size and speed. Indeed, once emitted, particles are independent entities whose properties can be altered as they progress along the link. Gates perform those alterations smoothly: the variable gets interpolated between the old and the new value over a short span centered on the gate, so as to avoid abrupt changes that would be aesthetically unpleasant and visually disturbing. Several examples in the next section use gates to alter particle variables. Variations in particle speed encode specific data attributes in Figure 3. Variations in particle opacity minimize legibility issues in Figure 4. Variations in particle color can be used, *e.g.*, for aesthetic purposes, or to visually illustrate the notion of transformation of what flows through the edges.
- *Tracks* enable particles to flow along multiple parallel paths inside the same link tunnel. All tracks that belong to the same link share the same emitter, for the sake of model simplicity. Nevertheless, the model allows for multiple links between two nodes, thus enabling different particle patterns on parallel tracks, as is done in Figure 5.

[2] A stroke pattern other than solid should be avoided, as it would visually interfere with the particles flowing along the link.

API

We have designed a Javascript API to map data attributes to any edge variable, including all three motion variables (pattern, frequency and speed). The following code fragment illustrates the simplicity of our API. It shows how a graph is loaded and laid out (lines 1-5), similar to D3 [9] or cola.js [18]. Lines 6-7 set the parameters controlling the particles' visual properties, while lines 8-10 set motion properties.

```
1  d3.json("data.json", function(json) {
2    var force = flownet.force("#aFrame", 800, 600, "#FFF", 0)
3                       .nodes(json.nodes)
4                       .links(json.links)
5                       .create_layout()
6                       .particles("color", "#CCC")
7                       .particles("size", 2)
8                       .particles("pattern", [0.0, 0.5, 0.75])
9                       .particles("frequency", 0.4)
10                      .particles("speed", 3);
11 });
```

Mapping these parameters to actual edge data attributes is achieved using anonymous functions, as shown below (d being a link instance, following the same convention as in D3). In this example, particle speed is set to be proportional ($\times 10$) to the value of edge attribute val on each link:

```
.particles("speed", function(d){return d.val * 10;})
```

Temporal patterns are specified as arrays of floats in $[0, 1[$. Each item in the array corresponds to a particle in the sequence defining the motif. Line 8 in the above code fragment defines a pattern made of three particles, fired by the emitter at the beginning (0.0), middle (0.5) and three quarters (.75) of each cycle. The actual timing depends on the pattern frequency for each edge. In this case, the pattern frequency being set to 0.4 Hz, the three particles are fired at 0s, 1.25s and 1.875s, repeating every 2.5s. Patterns can be assigned to individual edges based on any attribute, as shown below with a test of attribute cat:

```
.particles("pattern", function(d){
    if (d.cat == "dualLink"){return [0, .33, .66];}
    else {return [0, .2];}
})
```

In the example below, either one or two tracks are created, depending on the value of attribute cat for each edge:

```
force.tracks("count", function(d){return (d.cat == "dualLink") ? 2 : 1;});
```

Alterations to visual and motion attributes when passing through gates are specified by passing one additional parameter to function particles(). In the example below, particles are colored green on the first half of the link, and red on the second. Similarly, the speed of particles is 10 times the value of edge attribute val during the first 80% of their course, and increases to 20 times that value on the remaining 20%:

```
.particles("color", "#0F0", 0)
.particles("color", "#F00", .5)
.particles("speed", function(d){return d.val*10;}, 0)
.particles("speed", function(d){return d.val*20;}, .8);
```

Prototype Implementation

We have developed a prototype Javascript implementation of our model, called FlowNet. FlowNet can be used in conjunction with D3 to load and prepare the data for visualization, and graph layout libraries such as cola.js to compute node placements. Rendering takes place in an HTML5 `canvas` element using Three.js [15]. Using SVG for rendering would have enabled a tighter integration of FlowNet with D3, but our early tests showed that this solution would not have been able to handle large numbers of particles. Using WebGL enables us to take advantage of GPU rendering, vertex and fragment shaders being able to manage larger amounts of particles.

EXAMPLES OF USE

We illustrate different combinations of motion variables using subsets of the air traffic statistics dataset introduced earlier. The data are not about individual flights, but are rather aggregated statistics over all airlines on each route, thus forming a network that features a rich set of attributes for both nodes (airports) and links (flight routes). Animated versions of the examples shown here are available in the companion video and Website.[3] For the sake of clarity, we selected very small data subsets, which lend themselves better to static representations for inclusion in the paper. As stated earlier, motion variables should not be seen as systematic replacements for other visual variables such as color and stroke width, but rather as alternative visual variables that widen the space of possibilities and have their own strengths and weaknesses.

Encoding a single attribute

The simplest example consists of encoding a single edge attribute using a motion variable. Both numerical and categorical data can be encoded using motion. In the air traffic dataset, numerical attributes associated with each edge include the number of flights on the corresponding route, or the average departure delay on that route. Categorical attributes include the type of route (passenger transportation, mail, or freight).

Encoding a numerical attribute.
Numerical attributes can be directly mapped to speed or frequency. One guiding principle in the selection of which motion variable to choose in the design space is to pay attention to the semantics inherently conveyed by that variable. For instance, when seeking to encode the number of flights on a route, frequency is a more appropriate choice than speed, as users will tend to interpret a higher frequency, which entails a higher particle density on the link, as a larger number of flights. Conversely, speed may convey the notion of a more efficient flight route in terms of, *e.g.*, travel time, or passenger "throughput" as already hinted at in the Motivation section.

More complex attribute sets can be encoded using elaborate particle motion behaviors. The air traffic dataset contains information about the average departure and arrival delays on each route. As illustrated in Figure 3, this information can be encoded by inserting two gates on the link to vary the speed of particles as they progress from origin to destination. Routes with no delay on either side will feature particles travelling

[3] http://ilda.saclay.inria.fr/flownet

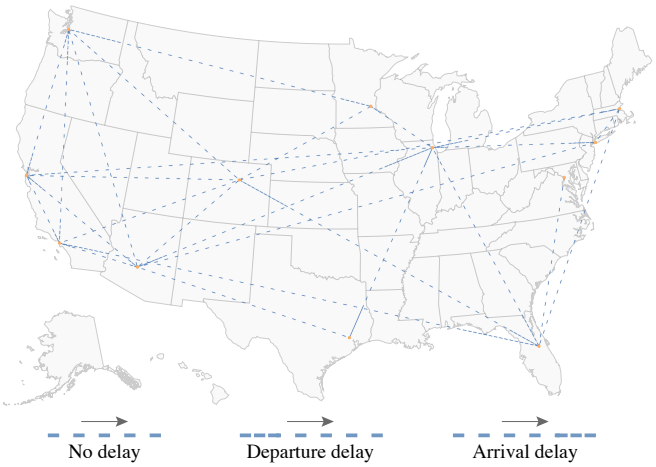

Figure 3. Particle speed mapped to departure and arrival delay.

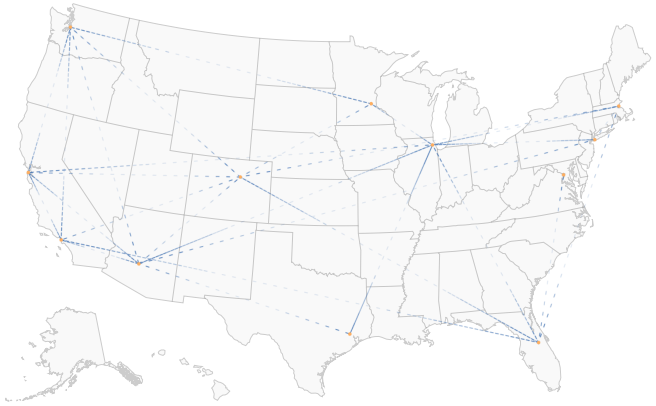

Figure 4. Same as in Figure 3, also reducing edge clutter by gradually fading particles out in the central portion of each edge.

at constant speed. Routes with departure delay will feature particles that start at a low speed, attaining their normal speed once they have travelled 20% of the link's length. On routes with arrival delay, particles decelerate once they reach 80% of the link's length.

Reducing Visual Complexity

Gates can also be used to reduce visual complexity in the diagram, for instance by removing some clutter. The node-link diagram from the previous example (Figure 3) features numerous crossing and overlapping edges, in part because of the constraints on the nodes, which are geolocated. Inspired by Becker *et al.*'s shortening lines [7], we used gates to fade particles away in the central portion of the links, as illustrated in Figure 4. Decreasing the particles' opacity in this manner helps reduce visual clutter. Here, the motion of particles helps track a particular link, in part thanks to the Gestalt law of common fate that contributes to apprehending the particles that flow on that edge as one coherent group.

Encoding a categorical attribute.

Categorical attributes are best encoded using variables that do not inherently convey an order, such as temporal pattern (which is akin to Morse code). For example, particles can be emitted according to different patterns depending on the route

Figure 5. Using tracks to show different route types between airports.

type (passenger, mail, freight). Difficulties will arise when an edge has more than one value for the same attribute. When using shape or color hue to encode categorical data, multiple values can be represented by, *e.g.*, merging the shapes, or creating a hatched pattern featuring all relevant color hues. Similarly, all relevant individual particle patterns can be juxtaposed to form a more elaborate pattern that is representative of this combination. However, as in the case of shape or color hue, this solution only works for a very small amount of values. It also requires a minimum edge length to be discernible, as the length of the pattern is equal to, and even slightly larger than, the sum of all individual patterns involved.

Encoding multiple attributes

Combining multiple attributes (one might want to also display, *e.g.*, the average number of passenger or cargo flights) can be achieved using motion variables in conjunction with other visual variables such as the particles' color or size. For combinations of numerical and categorical attributes, particle speed or frequency can encode the numerical attribute, while particle hue (color) encodes the categorical attribute. For combinations of numerical attributes, one can leverage visual variables such as size or opacity in conjunction with motion.

However, encoding multiple categories on a single edge can rapidly become visually complex. One possible solution is to use more than one track on an edge, each track being mapped to a different attribute. In Figure 5, the three tracks are associated with passengers, mail and freight, respectively. The category each track represents is visualized using a dual encoding: the track index (they are always ordered in the same way), and one of three particle patterns. Indeed the track index is likely not sufficient, considering that links can have different orientations. Then, the average number of flights for each of the three categories on each route is encoded using the particles' speed, which can thus be different on each track.

EXPERIMENTS

The above examples try to illustrate a set of representative encodings that use motion, but they say nothing about users' ability to perceive the values encoded in this manner. A few user studies of graph visualization techniques include one or more conditions that involve motion in the way animated edge textures do [34, 60]. However, to the best of our knowledge,

there are no empirical data on the number and type of motion patterns of edge particules that users can discern. Based on the model presented earlier, we start to explore the design space defined by animated edge textures, and provide initial empirical data about their potential for encoding information.

We report on an experiment that aims at evaluating the encoding potential of each motion variable *individually*. This is an initial study, and more studies will be necessary to fully understand the interplay between motion variables as well as with other visual variables, as discussed later. In our experiment, participants were presented with node-link diagrams whose links featured different animated edge textures. They were asked to group edges based on the motion of particles.

Our primary factor is the type of motion variable (*Motion_Variable* $\in$ {*Pattern*, *Speed*, *Frequency*}) that can take up to six different values (*i.e.*, participants have to identify up to six different groups of edges). *Pattern* is the pattern of particles flowing along the edge. *Speed* and *Frequency* are respectively the speed of particles and the frequency at which the source node emits the pattern. As mentioned above, our experimental design varies a single motion property at a time, keeping the other two properties set to their default value ($Pattern_0$, $Speed_0$ and $Frequency_0$). $Pattern_0$ is a single particle, *i.e.*, the simplest pattern. $Speed_0$ and $Frequency_0$ were chosen so as to ensure that there is always more than one occurence of the pattern visible on the smallest link that the diagrams used in the experiment can feature.[4]

We used one of two different strategies for choosing the different values to test, depending on the property considered. In the case of *Pattern*, there is no natural order between different rhythms, which makes it a categorical property. Inspired by previous work on rhythmic interaction [24], we chose the six different particle patterns shown in Figure 1. In the case of *Speed* and *Frequency*, which are continuous properties, we had to ensure a minimal *difference threshold* between consecutive levels, above their *Just Noticeable Difference* (JND) [20] in order to ensure that participants could perceive them as different. We first explain how we chose this difference threshold, and then report on our experimental design and observations.

Difference Threshold for *Speed* and *Frequency*

According to the Weber-Fechner law [20], the JND must be assessed as a constant proportion of the original stimulus I, meaning that detecting a change from an initially-high level requires a higher-amplitude change than detecting a change from an initially-low level: $JND = \Delta I / I$.

The notion of JND has been applied to various perceptual channels such as sound [54] and chromaticity [41]. Previous studies have considered perception of changes in speed or frequency, but those findings do not readily apply to the specific context of animated edge textures. For example, studies in experimental psychology have considered reaction time to a sudden change in velocity of a single stimulus [42]. This differs significantly from our context, where people must compare multiple stimuli (animated links in a diagram) that coexist on screen. Huber *et al.* [35] have studied users' ability to identify a target group within a larger group, based on its difference in velocity or flickering frequency relative to an otherwise-uniform group. However, in their experiments, the elements were simple squares that filled the display area. In a node-link diagram, edges are not uniform elements paving the space. They are distant from one another, and vary in both direction and size.

We ran a pilot study to get an approximation (accurate-enough for our purposes) of the JND at different reference values for both the *Speed* and *Frequency* motion variables. Six participants were asked to tell whether or not they perceived a difference between a source edge and a target edge, which varied in their orientation. At the beginning of each trial, both edges were animated with particles flowing at the same *Speed* (resp. *Frequency*). Participants had to press a key repeatedly until they saw a difference. Then they had to say whether the difference was an increase or a decrease. For both motion variables, we observed difference thresholds between 0.13 and 0.26, with no significant effect of the difference in orientation between the source and the target edges on the perceived difference. We then chose thresholds significantly higher than the observed JNDs (0.7 for *Speed*, 0.5 for *Frequency*) based on pilot tests on actual node-link diagrams, accounting for the fact that comparing particle frequencies and speeds in a node-link diagram featuring dozens of nodes and edges is necessarily more difficult than the comparison of a single pair of edges in an otherwise empty display.

Encoding Edge Attributes

This experiment followed a between-subject design, with three independent groups of participants testing three different motion variables: *Pattern*, *Speed* and *Frequency*. Our experiment software is available for illustration and replication purposes.[3]

Task and Procedure

The experimental task consisted in grouping edges according to the values of the tested *Motion_Variable* (several links could have the same texture assigned to them). While such a task is definitely not a realistic visual graph analysis task, we believe it provides a relevant operationalization for the purpose of this visual perception experiment, as it requires participants to perform visual comparisons between edges not only to identify how many different values exist in the graph, but also to perform pairwise comparisons to identify which edges belong to the same group.

As we aim at estimating the number of different values that each motion variable can encode, we want to find out whether users can group edges according to the different values of these properties present in the graph. Figure 6 shows a trial in the *Pattern* condition. The graph features six different *Pattern* values, meaning that participants should identify six groups of edges, provided they can actually discriminate all six values. A trial was divided into two phases.

[4] In the *Pattern* condition, $Speed_0$ is 6 mm.s^{-1}, and $Frequency_0$ is 0.3 Hz. In the *Speed* condition, $Frequency_0$ is 1.2 Hz. In the *Frequency* condition, $Speed_0$ is 7.5 mm.s^{-1}.
We also make nodes emit particles with a phase shift in order to prevent participants from discriminating frequencies among edges connected to a node simply by looking in its immediate vicinity.

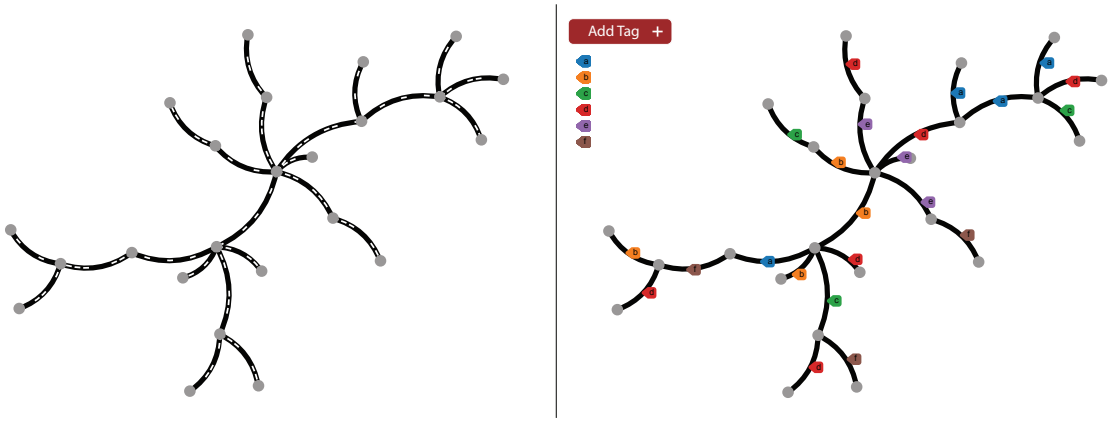

Figure 6. The second phase of our experimental task. The diagram on the left pane features animated edge textures, while the one on the right contains a copy of that diagram featuring static, solid edges. The right pane is interactive. It allows participants to create new tag types, which get added to the tag bar, and to tag individual edges. The background and edge colors have both been inverted in this screen capture: during the experiment, the background was black, and the edges were light gray.

In the first phase, participants had up to 45 seconds to tell how many different groups they perceived. They were encouraged to answer *as fast as they could*. They could press the space bar at any time before the timer expired. Either of these events (space bar pressed or the 45 seconds elapsed) made the graph disappear. Participants then had to input their answer in a text field. This task was an estimation task, as opposed to a subitization or counting task [29].

In the second phase, illustrated in Figure 6, participants had to actually identify these groups, creating the appropriate number of group tags and tagging each individual edge. Participants were instructed to be *as accurate as they could*. To avoid the tags interfering with the perception of the diagram, the display was split in two parts. The diagram on the left featured animated edge textures, while the one on the right contained a copy of that diagram featuring static, solid edges. Participants looked at the left-hand side diagram to identify groups, and used the right-hand side one to tag edges. They were able to create new tag types in the tag bar on-demand, using a button. Tagging an edge then entailed either dragging a tag from the bar and dropping it on that edge, or selecting a tag in the bar and clicking on the edge. The bar tag selection being persistent, the latter mechanism was especially useful to tag several edges in a row. Each edge could only hold one tag.

The operator insisted on the fact that it did not matter if the number of groups found in the second phase of a trial did not match that found in the first phase. The first phase rather aims at evaluating the potential of each *Motion_Variable* to convey a first impression, while the second step aims at capturing a more thorough analysis of the diagram (*i.e.*, where users have to tell whether two edges share the same animated texture or not in graphs featuring more or less diversity).

In addition to the *Motion_Variable* (*Pattern*, *Speed* and *Frequency*), our experiment also considered tasks of varying difficulty, which we operationalized using two factors.

First, we tested different graph sizes: *Graph_Size* $\in$ {LOW, MEDIUM, HIGH}. The node-link diagram layout was computed using D3's force-layout algorithm, with 8 nodes and 7 links for LOW, 16 nodes and 14 edges for MEDIUM, and 22 nodes and 21 links for HIGH. In order to avoid introducing too many factors in this first evaluation, we considered planar graphs only (the layouts did not contain any edge crossing).

Second, we varied the number of groups (*Group_Count*). Our purpose was to test whether increasing the number of different values of a motion variable diminishes users' ability to discriminate them. Factor *Group_Count* $\in$ {SMALL=2, MEDIUM=4, LARGE=6} corresponds to the number of different values that *Motion_Variable* could take. As already mentioned, we picked the six different rhythms shown in Figure 1, which we believed would be discriminable considering the range of edge sizes in our layouts (average: 3.18mm, min: 2.87mm, max: 3.45mm). Values of *Speed* and *Frequency* were computed as successive values right above the corresponding difference threshold (0.7 for *Speed*, 0.5 for *Frequency*) yielding, for each group count:

Group_Count	*Speed* values (mm.s^{-1})
SMALL	$G_1 = 3.3, G_2 = 5.6$
MEDIUM	$G_1 = 3.3, G_2 = 5.6, G_3 = 9.5, G_4 = 16.2$
LARGE	$G_1 = 3.3, G_2 = 5.6, G_3 = 9.5, G_4 = 16.2, G_5 = 27.5,$ $G_6 = 46.7$

Group_Count	*Frequency* values (Hz)
SMALL	$G_1 = 0.3, G_2 = 0.5$
MEDIUM	$G_1 = 0.3, G_2 = 0.5, G_3 = 0.7, G_4 = 1.0$
LARGE	$G_1 = 0.3, G_2 = 0.5, G_3 = 0.7, G_4 = 1.0, G_5 = 1.5,$ $G_6 = 2.3$

We chose to limit the maximum number of different values to 6 as a higher number of conditions would have resulted in an intractable experiment design. We ran pilot studies to identify an upper bound, which showed that differentiating 6 values was already challenging. In addition, going far beyond 6 values would have resulted in overly low-or-high speeds and frequencies, that would have been of little practical value.

Our three factors were tested following a mixed experiment design. To keep a reasonable experiment length (i.e., $\sim$ an hour), *Motion_Variable* was tested as a between-subject factor, each participant seeing only one of the three conditions. *Group_Count* and *Graph_Size* were tested as within-subject factors, all participants seeing all *Group_Count* $\times$ *Graph_Size* conditions. In order to get familiar with the task and the interactive tagging technique, participants started with 3 practice tri-

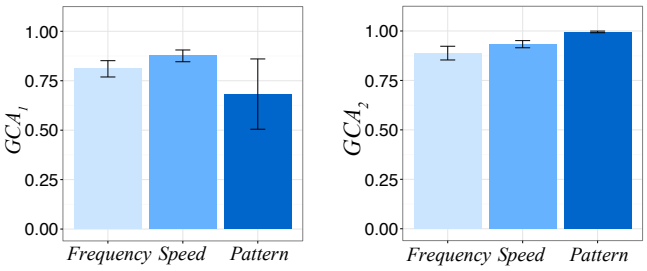

Figure 7. (Left) Accuracy in *estimating* the number of groups in the first phase; (Right) Accuracy in *counting* the number of different groups in the second phase. Error bars represent the 95% confidence interval.

als on small graphs featuring 2, 4, and 6 different groups. Then, each measured trial was replicated twice. In total, each participant had to complete 18 trials for analysis (3 *Group_Count* $\times$ 3 *Graph_Size* $\times$ 2 replications), presented in a random order. At the end of the experiment, participants filled in a short questionnaire asking them about the strategies that they used to identify the different groups of edges.

Participants & Apparatus

Thirty six volunteers (18 female), aged 21 to 47 year-old (average 30.8, median 29), participated in the experiment. We conducted the experiment on a PC Dell Precision T5500, equipped with an Intel Xeon Quad Core processor, 8GB RAM, and an NVidia Quadro 2000 graphics card driving a 23" LCD FullHD 1080p monitor (1920x1080, 96 dpi). The experiment software was developed in JavaScript using node.js [55] and the animated edge texture library described earlier.

Results

Our experimental software collected the estimation of the number of groups that users reported in the first phase, GC_1 (which stands for Group Count in 1st phase). It also recorded the full set of mappings <tag, edge> of the 2nd phase, from which it derived GC_2, the number of groups identified in that phase. We considered different measures for analyzing the collected data. We first computed the *Group Count Accuracy* (*GCA*) in both phases as follows:

GCA_1 = 1 - ($|GC_1$ - *Group_Count*$|$) / *Group_Count*, and

GCA_2 = 1 - ($|GC_2$ - *Group_Count*$|$) / *Group_Count*.

However, GCA_2 alone is not sufficient to account for the grouping accuracy, as the number of tag groups can be accurate while having some edges misclassified. For each group, we thus also computed a measure of its accuracy using the Jaccard coefficient [38]. This coefficient estimates similarity between finite sample sets using the ratio between the size of the intersection and the size of the union of the sample sets. As we cannot know *a priori* what actual groups users were tagging, we try to associate each group G_i with the most similar tag group. The *Grouping Accuracy* of a group G_i, $GA(G_i)$, is thus the maximum of all Jaccard coefficients computed on each pair made of G_i and one of the groups T_j ($1 \leq j \leq GC_2$) that the participant has tagged:

$$GA(G_i) = \max_{1 \leq j \leq GC_2} \frac{count(T_j \bigcap G_i)}{count(T_j \bigcup G_i)} \qquad 1 \leq i \leq Group\_Count$$

In the first phase, we observe an average accuracy GCA_1 of 0.81 for *Frequency*, 0.88 for *Speed* and 0.68 for *Pattern*. Participants seem to have more trouble estimating the number of groups when the discriminating motion variable is *Pattern*. However, an analysis of variance reveals that the difference between the three motion variables on GCA_1 is not significant ($p = 0.7$). In the second phase, the average accuracy GCA_2 increases to 0.88 for *Frequency*, 0.93 for *Speed* and 0.99 for *Pattern*. Here, the effect of *Motion_Variable* is significant on GCA_2 ($F_{2,33} = 3.5$, $p = 0.04$, $\eta^2_G = 0.17$), with each pair of conditions significantly differing according to pairwise t-tests.[5] Figure 7 illustrates these results. An interesting observation is that while *Pattern* performs relatively poorly as a means to get a quick *estimate* of the number of groups, it actually outperforms both other variables when the task is to more precisely *count* those groups. We tentatively explain this observation based on the fact that particle frequency and speed can be quickly perceived, while the identification of different patterns requires more cognitive processing. When trying to get an estimate under time pressure, different *Frequencies* or different *Speed*s will be easier to perceive, while when trying to get a precise count without such time pressure, *Pattern* will provide a cognitively more demanding, but more reliable, solution. Answers to the final questionnaire actually support this interpretation: participants reported experiencing difficulty memorizing patterns. On the opposite, in *Speed* and *Frequency* conditions, some of them reported using the simple strategy of looking only at the spacing between two particles to discriminate different values.

Figure 8 provides a detailed view of how the different motion variables are robust against increasing difficulty. Even if an analysis of variance of *Graph_Size* $\times$ *Group_Count* on GCA_2 does not reveal any significant difference in any of the *Motion_Variable* conditions, some edge properties seem to scale better to complex tasks (large graph, high number of groups) than others. *Pattern* and *Speed* seem to scale quite well with both *Graph_Size* and *Group_Count*, while users' accuracy with *Frequency* appears to drop down on average as the graph becomes larger. However, our GCA_2 measure makes by definition the error cost inversely proportional to the number of groups (*e.g.*, a misclassified edge impacts the accuracy three times more when *Group_Count*=SMALL than when *Group_Count*=LARGE). We complement our error analysis with an absolute measure of errors, that is the number of misclassified edges *NME*. An analysis of variance of *Graph_Size* $\times$ *Group_Count* on *NME* revealed more contrasted differences. For *Pattern*, there is still no significant difference between conditions. But, when *Motion_Variable=Frequency*, *Graph_Size* has a significant effect on *NME* ($F_{2,22} = 9$, $p = 0.001$, $\eta^2_G = 0.27$), with *NME* significantly increasing with *Graph_Size*. Finally, when *Motion_Variable=Speed*, all *Graph_Size*, *Group_Count* and *Graph_Size* $\times$ *Group_Count* interaction effects are significant on *NME*.[6] Pairwise t-tests reveal that the effect of *Group_Count* is significant only when

[5] We get the same results with or without Bonferroni correction.

[6] *Graph_Size*: $F_{2,22} = 10$, $p < 0.0001$, $\eta^2_G = 0.27$; *Group_Count*: $F_{2,22} = 10$, $p < 0.0001$, $\eta^2_G = 0.2$; and *Graph_Size*$\times$*Group_Count*: $F_{4,44} = 6$, $p < 0.0001$, $\eta^2_G = 0.16$

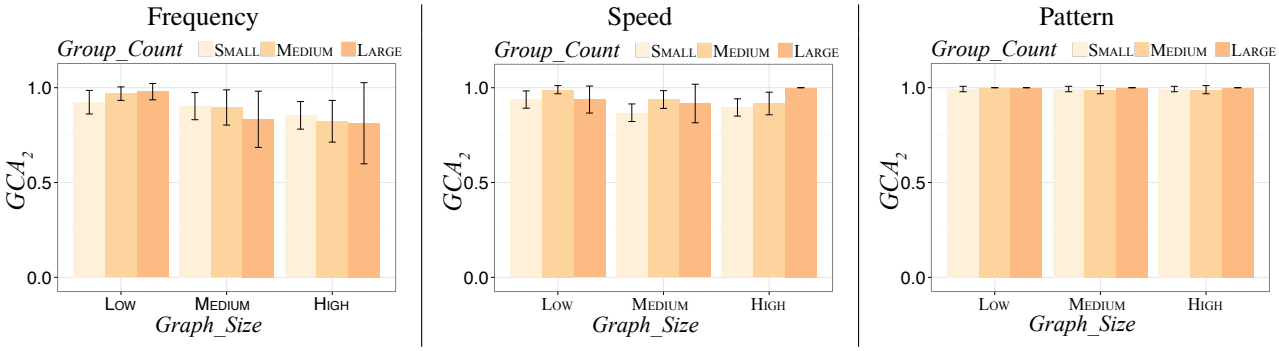

Figure 8. Average accuracy in second step (GCA_2) per *Graph_Size* × *Group_Count*. Error bars represent the 95% confidence interval.

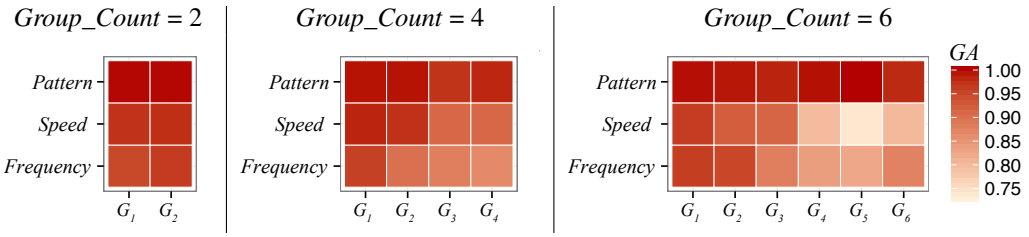

Figure 9. Confusion between groups per *Group_Count* condition. In a matrix, a row corresponds to a *Motion_Variable* condition, and a column to a group of edges. The color of a cell depends on the average accuracy of that group during the experiment (darker is better).

Graph_Size = HIGH. For SMALL- and MEDIUM-sized graphs, the number of misclassified edges does not significantly increase with the number of groups.

Figure 9 helps understand where the confusion between groups comes from. In those matrices, a cell represents a group of edges G_i, and the color of that cell is proportional to the average *Grouping Accuracy* of that group $GA(G_i)$. The darker a cell, the most accurate a group is. We can observe that increasing the number of groups for *Frequency* and *Speed* makes the accuracy drop down. We can also observe that participants tend to make more confusions between high values of *Frequency* and *Speed*, the accuracy being lower starting from G_4. It is important to remember that we increase *Group_Count* by adding values at the end of the range, which entails that a group G_i, when it exists in two different *Group_Count* conditions, is exactly the same in those two conditions.

In summary, *Pattern* seems to have good potential as a means to encode edge attributes. In this condition, participants in our experiment were able to identify the different edge groups almost perfectly, no matter the task difficulty. Participants also performed well with *Speed* and *Frequency*, but these properties seem to suffer more from an increase in the number of links. When *Frequency* or *Speed* are used to encode edge data attributes, users may experience some difficulty with large networks, and may experience difficulty discriminating between the higher frequencies and between the higher speeds.

FOLLOW-UP STUDY

Answers to the questionnaire in the study reported above revealed that some participants had relied on the spacing between particles to compare different values of *Speeds* or *Frequencies*. Because *Speed* and *Frequency* can be manipulated independently in our model, modifying the value of one of these motions variables impacts the spacing between particles. Because this latter property can also be perceived on a static representation of the texture (*e.g.*, on a snapshot of the graph), we also wanted to test what users can discriminate using motion dynamics only. We ran a follow-up study to test a fourth value for *Motion_Variable*, that we call *FASpeed*, for Frequency-Adjusted Speed. *FASpeed* adapts the frequency as speed varies so as to preserve a constant spacing between particles across the different *Speed* conditions.

As in the main experiment, we ran a pilot study to estimate the *Just Noticeable Difference* in a comparison task between a pair of edges. This informed the choice of consecutive values of *FASpeed*, computed using a threshold of 0.9:

Group_Count	*FASpeed* values (mm.s^{-1})
SMALL	$[G_1 = 3.75, G_2 = 7.125]$
MEDIUM	$[G_1 = 3.75, G_2 = 7.125, G_3 = 13,5375, G_4 = 25,725]$
LARGE	$[G_1 = 3.75, G_2 = 7.125, G_3 = 13,5375, G_4 = 25,725,$ $G_5 = 48,8775, G_6 = 93]$

Participants & Apparatus

Fifteen volunteers (3 female), aged 23 to 37 year-old (average 27.87, median 28), participated in this experiment. None of them was involved in the previous study. The experiment was run on a 2.7GHz Intel Core i5 Macbook Pro, equipped with 8GB RAM and an Intel Iris Graphics 6100 1536 Mo driving a 27" LCD monitor (2560x1440, 100 dpi).

Task & Procedure

Participants were exposed to the *Motion_Variable=FASpeed* condition only. As before, *Group_Count* and *Graph_Size* were tested using a within-subject design with two replications, with participants seeing a total of 18 trials for analysis. Trials were presented in a random order, after 3 practice trials. At the end of the experiment, participants filled in a short questionnaire about their strategies for grouping edges.

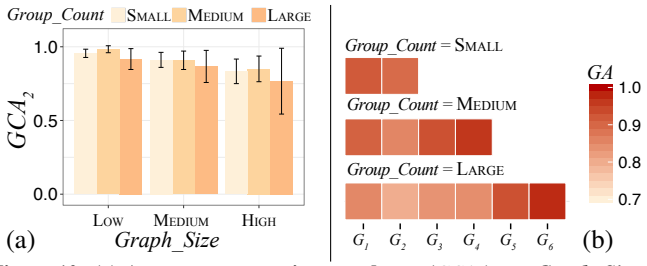

Figure 10. (a) Average accuracy in second step (GCA_2) per *Graph_Size* × *Group_Count*. (b) Confusion between groups per *Group_Count*.

Results

We observe a similar trend to what we have observed in the previous experiment: accuracy increases between the 1^{st} and the 2^{nd} phases, with $GCA_1 = 0.82$ and $GCA_2 = 0.89$ on average. *Group_Count* does not have a significant effect on these two measures, suggesting that users can discriminate up to six different values. However, participants experienced more difficulty with large graphs than with small ones. The effect of *Graph_Size* is significant on both $GCA_1 (F_{2,28} = 8.86, p = 0.001, \eta^2_G = 0.08)$ and $GCA_2 (F_{2,28} = 9.02, p = 0.001, \eta^2_G = 0.11)$, with pairwise t-tests revealing that all *Graph_Size* conditions were significantly different from each other (LOW > MEDIUM > HIGH). Figure 10-(a) illustrates these results for GCA_2.

The effect of *Graph_Size* on the total number of misclassified edges (*NME*) is larger ($F_{2,28} = 15.7, p = 0.00001, \eta^2_G = 0.25$) than that of *Group_Count* ($F_{2,28} = 3.5, p = 0.04, \eta^2_G = 0.08$). This is consistent with what we observed about *Speed* and *Frequency* in the previous experiment, where the number of misclassified edges grew with *Graph_Size*. However, as opposed to the *Speed* condition in the previous experiment, confusion matrices (Figure 10-(b)) illustrate that confusions are more frequent between low values than high values of *FASpeed*.

In summary, participants were able to discriminate up to six different values based exclusively on the motion dynamics of flowing particles, with an accuracy of $\sim 90\%$. However, we also observe that the size of the graph impacts participants' accuracy. Large graphs involve comparisons between edges that can be far away from each other, and we observe that comparing two edges based on their dynamics is a difficult task when they do not both fall simultaneously in the center of visual attention. This is in line with participants' comments at the end of the experiment, where they reported having difficulty to "recall" the speed of the two edges that they wanted to compare when they were distant from each other.

DISCUSSION AND FUTURE WORK

Recent work on confluent drawings [2] recently identified *"particle animations simulating flow along edges"* as an intriguing technique *"for future research in visualizing flow and propagation in networks"*. Our work is a step in this direction. We introduce a design space for animated edge textures, providing a framework to guide the exploration and evaluation of motion variables as a means to visually encode edge data attributes in node-link diagrams. That design space is quite large, especially when considering all possible combinations of motion variables and other visual variables that define the appearance of both the particles and the links themselves.

Beyond the possibilities afforded by animated edge textures in terms of visual design, a central question is that of the actual effectiveness of motion variables for the visual encoding of data values. The two perception experiments reported in this paper provide initial quantitative data about each motion variable studied in isolation. These results are encouraging, suggesting that each variable in our model has some potential in practice. But these results do not generalize to visual mappings involving multiple motion variables. More empirical evidence needs to be gathered to fully understand the interplay between variables and comprehensively evaluate the potential of animated edge textures. In particular, it is not yet clear how motion variables interact (and potentially interfere) with one another. Conditions that will have to be considered in future work include: the perception of conjunctions of motion variables (*e.g.*, particle pattern and speed); or conjunctions of motion and other visual variables (*e.g.*, particle color and speed, taking into account prior work on the role of color in motion perception [64]). Answering such questions will necessarily require many more experiments, that might very well reveal effects similar to what happens when, *e.g.*, combining two pre-attentive channels like color and shape [56]: while both are pre-attentive when considered in isolation, they are no longer pre-attentive when combined in a conjunctive visual search. Variations in motion properties that occur as particles progress along links such as, *e.g.*, the changes in velocity illustrated in Figure 3, should also be studied.

Other relevant factors to study include the type of graphs, their layout, and a larger range of sizes than what was tested here. For instance, edge crossings (Figures 3 and 4) are likely to have an impact on the perception of particle motion when visualizing non-planar graphs. Different topologies (including graphs featuring bidirectional links) should also be tested, varying graph complexity as was done here. Indeed, both graph topology and size will impact overall readability and thus, possibly, the perception of motion patterns. A related research question is that of which graph layout and edge routing techniques are best coupled with animated edge textures. For instance, confluent drawings [2] feature edge bundling that is likely to play well with particle-based animations, but this remains to be shown empirically. Even if considering less elaborate layouts, some factors will necessarily impact, and impose limitations on, the use of animated edge textures. For instance, the choice of particle pattern and speed depends to some extent on the length of edges: long patterns may not fit in their entirety on small edges; and similarly, fast particles might be difficult to spot when they travel short distances.

As hinted at above, we have barely scratched the surface of all questions raised by the use of motion to encode data in node-link diagrams. We make our API, as well as the code of our experimental setup, publicly available,[3] in order to encourage further exploration of this design space by the community, for the purpose of both creating novel visual designs and conducting empirical studies. The model itself could actually also be extended. For instance, we have started investigating another particle property, that controls the stability of its trajectory along the track. Making particles wiggle could be useful to, *e.g.*, encode uncertainty or variability in the data.

REFERENCES

1. Daniel Archambault and Helen C. Purchase. 2016. On the effective visualisation of dynamic attribute cascades. *Information Visualization* 15, 1 (2016), 51–63. DOI: http://dx.doi.org/10.1177/1473871615576758
2. B. Bach, N. Henry Riche, C. Hurter, K. Marriott, and T. Dwyer. 2017. Towards Unambiguous Edge Bundling: Investigating Confluent Drawings for Network Visualization. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 541–550. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598958
3. Lyn Bartram. 1997. Perceptual and Interpretative Properties of Motion for Information Visualization. In *Proceedings of the 1997 Workshop on New Paradigms in Information Visualization and Manipulation (NPIV '97)*. ACM, New York, NY, USA, 3–7. DOI: http://dx.doi.org/10.1145/275519.275520
4. Lyn Bartram and Colin Ware. 2002. Filtering and brushing with motion. *Information Visualization* 1, 1 (2002), 66–79. DOI: http://dx.doi.org/10.1057/palgrave.ivs.9500005
5. Lyn Bartram and Miao Yao. 2008. Animating causal overlays. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 751–758. DOI: http://dx.doi.org/10.1111/j.1467-8659.2008.01204.x
6. Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. 2014. The State of the Art in Visualizing Dynamic Graphs. In *EuroVis - STARs*, R. Borgo, R. Maciejewski, and I. Viola (Eds.). The Eurographics Association. DOI: http://dx.doi.org/10.2312/eurovisstar.20141174
7. R. A. Becker, S. G. Eick, and A. R. Wilks. 1995. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics* 1, 1 (Mar 1995), 16–28. DOI:http://dx.doi.org/10.1109/2945.468391
8. Jorik Blaas, Charl Botha, Edward Grundy, Mark Jones, Robert Laramee, and Frits Post. 2009. Smooth Graphs for Visual Exploration of Higher-Order State Transitions. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (Nov. 2009), 969–976. DOI: http://dx.doi.org/10.1109/TVCG.2009.181
9. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. DOI: http://dx.doi.org/10.1109/TVCG.2011.185
10. Stefan Buschmann, Matthias Trapp, and Jürgen Döllner. 2016. Animated visualization of spatial–temporal trajectory data for air-traffic analysis. *The Visual Computer* 32, 3 (2016), 371–381. DOI: http://dx.doi.org/10.1007/s00371-015-1185-9
11. Wei Chen, Fangzhou Guo, and Fei-Yue Wang. 2015. A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems* 16, 6 (2015), 2970–2984. DOI: http://dx.doi.org/10.1109/TITS.2015.2436897
12. Fanny Chevalier, Nathalie Henry Riche, Catherine Plaisant, Amira Chalbi, and Christophe Hurter. 2016. Animations 25 Years Later: New Roles and Opportunities. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '16)*. ACM, New York, NY, USA, 280–287. DOI: http://dx.doi.org/10.1145/2909132.2909255
13. Owen Cornec and Romain Vuillemot. 2015. Visualizing the Scale of World Economies. In *VisWeek 2015 Electronic Conference Proceedings - Poster*.
14. Pedro Cruz. 2015. Wrongfully right: applications of semantic figurative metaphors in information visualization. In *IEEE VIS Arts Program (VISAP)*. 14–21.
15. Jos Dirksen. 2015. *Learning Three.js – the JavaScript 3D Library for WebGL*. Packt Publishing Ltd.
16. Jon Driver and Peter McLeod. 1992. Reversing Visual Search Asymmetries With Conjunctions of Movement and Orientation. *Journal of Experimental Psychology: Human Perception and Performance* 18, 1 (1992), 22–33. DOI:http://dx.doi.org/10.1037/0096-1523.18.1.22
17. Jon Driver, Peter McLeod, and Zoltan Dienes. 1992. Motion coherence and conjunction search: Implications for guided search theory. *Perception & Psychophysics* 51, 1 (1992), 79–85. DOI: http://dx.doi.org/10.3758/BF03205076
18. Tim Dwyer. 2013. cola.js. http://marvl.infotech.monash.edu/webcola/ - Last accessed 2017-09-08. (2013).
19. R. Etemadpour, P. Murray, and A. G. Forbes. 2014. Evaluating density-based motion for big data visual analytics. In *2014 IEEE International Conference on Big Data (Big Data)*. 451–460. DOI: http://dx.doi.org/10.1109/BigData.2014.7004262
20. Gustav Fechner. 1966. *Elements of psychophysics. Vol. I.* American Psychological Association.
21. Chao Feng, Lyn Bartram, and Bernhard E. Riecke. 2014. Evaluating Affective Features of 3D Motionscapes. In *Proceedings of the ACM Symposium on Applied Perception (SAP '14)*. ACM, New York, NY, USA, 23–30. DOI:http://dx.doi.org/10.1145/2628257.2628264
22. Danyel Fisher. 2010. Animation for visualization: opportunities and drawbacks. In *Beautiful Visualization*. O'Reilly Media, Chapter 19, 329–352.
23. Steven L. Franconeri and Daniel J. Simons. 2003. Moving and looming stimuli capture attention. *Perception & Psychophysics* 65, 7 (2003), 999–1010. DOI: http://dx.doi.org/10.3758/BF03194829
24. Emilien Ghomi, Guillaume Faure, Stéphane Huot, Olivier Chapuis, and Michel Beaudouin-Lafon. 2012. Using rhythmic patterns as an input method. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1253–1262. DOI: http://dx.doi.org/10.1145/2207676.2208579

25. John A. Greenwood and Mark Edwards. 2009. The detection of multiple global directions: Capacity limits with spatially segregated and transparent-motion signals. *Journal of Vision* 9, 1 (2009), 40. DOI: http://dx.doi.org/10.1167/9.1.40

26. Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz. 2015. A Survey of Multi-faceted Graph Visualization. In *Eurographics Conference on Visualization (EuroVis) - STARs*, R. Borgo, F. Ganovelli, and I. Viola (Eds.). The Eurographics Association. DOI: http://dx.doi.org/10.2312/eurovisstar.20151109

27. S. Haroz, K. L. Ma, and K. Heitmann. 2008. Multiple Uncertainties in Time-Variant Cosmological Particle Data. In *IEEE Pacific Visualization Symposium*. 207–214. DOI: http://dx.doi.org/10.1109/PACIFICVIS.2008.4475478

28. Steve Haroz and David Whitney. 2010. Temporal Thresholds for Feature Detection in Flow Visualization. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization (APGV '10)*. ACM, 163–163. DOI: http://dx.doi.org/10.1145/1836248.1836285

29. Steve Haroz and David Whitney. 2012. How Capacity Limits of Attention Influence Information Visualization Effectiveness. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2402–2410. DOI: http://dx.doi.org/10.1109/TVCG.2012.233

30. Fritz Heider and Marianne Simmel. 1944. An experimental study of apparent behavior. *The American Journal of Psychology* 57, 2 (1944), 243–259. DOI: http://dx.doi.org/10.2307/1416950

31. Nathalie Henry Riche, Tim Dwyer, Bongshin Lee, and Sheelagh Carpendale. 2012. Exploring the Design Space of Interactive Link Curvature in Network Diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*. ACM, New York, NY, USA, 506–513. DOI: http://dx.doi.org/10.1145/2254556.2254652

32. Uta Hinrichs, Simon Butscher, Jens Müller, and Harald Reiterer. 2016. Diving in at the deep end: the value of alternative in-situ approaches for systematic library search. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4634–4646. DOI:http://dx.doi.org/10.1145/2858036.2858549

33. J Hohnsbein and S Mateeff. 1998. The time it takes to detect changes in speed and direction of visual motion. *Vision Research* 38, 17 (1998), 2569 – 2573. DOI: http://dx.doi.org/10.1016/S0042-6989(98)00014-5

34. Danny Holten, Petra Isenberg, Jarke J. van Wijk, and Jean-Daniel Fekete. 2011. An Extended Evaluation of the Readability of Tapered, Animated, and Textured Directed-edge Representations in Node-link Graphs. In *Proceedings of the 2011 IEEE Pacific Visualization Symposium (PacifiVis '11)*. IEEE Computer Society, 195–202. DOI: http://dx.doi.org/10.1109/PACIFICVIS.2011.5742390

35. Daniel E Huber and Christopher G Healey. 2005. Visualizing data with motion. In *Visualization, 2005. VIS 05. IEEE*. IEEE, 527–534. DOI: http://dx.doi.org/10.1109/VISUAL.2005.1532838

36. M. Itoh, D. Yokoyama, M. Toyoda, Y. Tomita, S. Kawamura, and M. Kitsuregawa. 2016. Visual Exploration of Changes in Passenger Flows and Tweets on Mega-City Metro Network. *IEEE Transactions on Big Data* 2, 1 (March 2016), 85–99. DOI: http://dx.doi.org/10.1109/TBDATA.2016.2546301

37. Richard B Ivry and Asher Cohen. 1992. Asymmetry in visual search for targets defined by differences in movement speed. *Journal of Experimental Psychology: Human Perception and Performance* 18 (1992), 1045–1045. DOI: http://dx.doi.org/10.1037/0096-1523.18.4.1045

38. Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New phytologist* 11, 2 (1912), 37–50. DOI: http://dx.doi.org/10.1111/j.1469-8137.1912.tb05611.x

39. M. Lockyer, L. Bartram, and B. E. Riecke. 2011. Simple Motion Textures for Ambient Affect. In *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging (CAe '11)*. ACM, New York, NY, USA, 89–96. DOI: http://dx.doi.org/10.1145/2030441.2030461

40. M Lockyer, L Bartram, T Schiphost, and K Studd. 2016. Enhancing Visualization with Expressive Motion. *Electronic Imaging* 2016, 16 (2016), 1–6. DOI:http://dx.doi.org/10.2352/ISSN.2470-1173.2016.16.HVEI-140

41. David L MacAdam. 1942. Visual sensitivities to color differences in daylight. *Journal of the Optical Society of America* 32, 5 (1942), 247–274. DOI: http://dx.doi.org/10.1364/JOSA.32.000247

42. S Mateeff, G Dimitrov, and J Hohnsbein. 1995. Temporal thresholds and reaction time to changes in velocity of visual motion. *Vision research* 35, 3 (1995), 355–363. DOI:http://dx.doi.org/10.1016/0042-6989(94)00130-E

43. Peter McLeod, Jon Driver, and Jennie Crisp. 1988. Visual search for a conjunction of movement and form is parallel. *Nature* 332, 6160 (03 1988), 154–155. DOI: http://dx.doi.org/10.1038/332154a0

44. Tony McLoughlin, Robert S Laramee, Ronald Peikert, Frits H Post, and Min Chen. 2010. Over Two Decades of Integration-Based, Geometric Flow Visualization. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 1807–1829. DOI: http://dx.doi.org/10.1111/j.1467-8659.2010.01650.x

45. A. Michotte. 1963. *The perception of causality*. Basic Books.

46. Till Nagel, Christopher Pietsch, and Marian Dörk. 2016. Staged Analysis: From Evocative to Comparative Visualizations of Urban Mobility. In *IEEE VIS Arts Program (VISAP)*. 23–30.

47. Ken Nakayama and Gerald H. Silverman. 1986. Serial and parallel processing of visual feature conjunctions. *Nature* 320, 6059 (03 1986), 264–265. DOI: http://dx.doi.org/10.1038/320264a0

48. Guy A. Orban, Frank Van Calenbergh, Bart De Bruyn, and Hugo Maes. 1985. Velocity discrimination in central and peripheral visual field. *Journal of the Optical Society of America A* 2, 11 (Nov 1985), 1836–1847. DOI: http://dx.doi.org/10.1364/JOSAA.2.001836

49. Emmanuel Pietriga and Denis Barkats. 2014. Antenna dance at ALMA. http://www.almaobservatory.org/en/announcement/antenna-dance-at-alma/ - Last accessed 2018-01-04. (2014).

50. Roeland Scheepens, Christophe Hurter, Huub Van De Wetering, and Jarke J Van Wijk. 2016. Visualization, selection, and analysis of traffic flows. *IEEE transactions on visualization and computer graphics* 22, 1 (2016), 379–388. DOI: http://dx.doi.org/10.1109/TVCG.2015.2467112

51. Jason M. Scimeca and Steven L. Franconeri. 2015. Selecting and tracking multiple objects. *Wiley Interdisciplinary Reviews: Cognitive Science* 6, 2 (2015), 109–118. DOI:http://dx.doi.org/10.1002/wcs.1328

52. R. Sekuler, S.N.J. Watamaniuk, and R. Blake. 2004. Perception of Visual Motion. In *Stevens' Handbook of Experimental Psychology: Vol. 1 Sensation and perception (3rd edition)*. Wiley & Sons, 121– 176.

53. Tatiana Tekušová, Jörn Kohlhammer, Slawomir J. Skwarek, and Galina V. Paramei. 2008. Perception of Direction Changes in Animated Data Visualization. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization (APGV '08)*. ACM, 205–205. DOI: http://dx.doi.org/10.1145/1394281.1394331

54. Kim Thomas. 2007. Just noticeable difference and tempo change. *Journal of Scientific Psychology* 2 (2007), 14–20.

55. Stefan Tilkov and Steve Vinoski. 2010. Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing* 14, 6 (2010), 80–83. DOI: http://dx.doi.org/10.1109/MIC.2010.145

56. Anne Treisman. 1985. Preattentive Processing in Vision. *Comput. Vision Graph. Image Process.* 31, 2 (Aug. 1985), 156–177. DOI: http://dx.doi.org/10.1016/S0734-189X(85)80004-9

57. Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. 2002. Animation: Can It Facilitate? *Int. J. Hum.-Comput. Stud.* 57, 4 (Oct. 2002), 247–262. DOI: http://dx.doi.org/10.1006/ijhc.2002.1017

58. Tatiana von Landesberger, Simon Diel, Sebastian Bremm, and Dieter W Fellner. 2015. Visual analysis of contagion in networks. *Information Visualization* 14, 2 (2015), 93–110. DOI: http://dx.doi.org/10.1177/1473871613487087

59. T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.-D. Fekete, and D.W. Fellner. 2011. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum* 30, 6 (2011), 1719–1749. DOI: http://dx.doi.org/10.1111/j.1467-8659.2011.01898.x

60. Colin Ware and Robert Bobrow. 2004. Motion to Support Rapid Interactive Queries on Node-link Diagrams. *ACM Trans. Appl. Percept.* 1, 1 (July 2004), 3–18. DOI: http://dx.doi.org/10.1145/1008722.1008724

61. Colin Ware, Joseph Bonner, William Knight, and Rod Cater. 1992. Moving icons as a human interrupt. *International Journal of Human-Computer Interaction* 4, 4 (1992), 341–348. DOI: http://dx.doi.org/10.1080/10447319209526047

62. Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. 2002. Cognitive Measurements of Graph Aesthetics. *Information Visualization* 1, 2 (2002), 103–110. DOI: http://dx.doi.org/10.1057/palgrave.ivs.9500013

63. Martin Wattenberg. 2006. Visual Exploration of Multivariate Graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 811–819. DOI: http://dx.doi.org/10.1145/1124772.1124891

64. D. Weiskopf. 2004. On the role of color in the perception of motion in animated visualizations. In *IEEE Visualization 2004*. 305–312. DOI: http://dx.doi.org/10.1109/VISUAL.2004.73